Homework 7

Out: 11.14.16
Due: 11.28.16

1. [Graph algorithms, 30 Points]
   a. Give an O(V+E) algorithm for computing the number of connected components of an undirected graph G(V,E). For each connected component, your algorithm should provide the number of vertices and edges.

   b. An independent set of an undirected graph G(E,V) is a subset I of V such that no two vertices in I are adjacent. That is, if u and v are in I, then (u,v) is not in E. Give an efficient algorithm that computes a maximal independent set for a graph G. (That is, if we add any additional vertex to this set, it will no longer be independent.) What is the runtime of your algorithm?

   c. The following greedy algorithm finds a shortest path from vertex *start* to vertex *goal* in a given connected graph:

      1. Initialize: *path* = {*start*}, *VisitedVertices* = {*start*}.
      2. If (*start* = *goal*), return *path* and exit.
      3. Find the edge (*start, v*) of minimum weight such that *v* is adjacent to *start* and *v* is not in *VisitedVertices.*
      4. Add *v* to *path*.
      5. Add *v* to *VisitedVertices*.
      6. Set *start* = *v*, and go to step 2.

      Does this algorithm always find a shortest path from *start goal*? Explain why it works, or give a counter example.

   d. Computer networks should avoid single points of failure, that is, network nodes that can disconnect the network if they fail. A connected graph G is bi-connected if it contains no vertex whose removal would divide G into two or more connected components. Give an O(V+E) algorithm for adding at most V edges to a connected graph G, with $V \geq 3$ vertices and $E \geq V-1$ edges, to guarantee that G is bi-connected. Specify any data structures that you are using.

2. [MST, 15 points]
   In graph G(V,E), all edge weights are integers in the range 1..W, for some constant W.
   a. How fast can Prim's algorithm run on G? Explain.
   b. How fast can Kruskal's algorithm run on G, assuming a bunch of linked lists with a weighted heuristic? Explain, and specify any assumptions that you are making.

3. [MST, 15 points]
Each of the following algorithms takes a connected weighted graph G(V,E) as input, and returns a set of edges T. For each algorithm, either explain why T is a minimum spanning tree or give a counter-example.

   a. MAYBE-MST-A(G)

```
T = empty
for each edge e, taken in arbitrary order
        if T ∪ {e} has no cycles
                T = T ∪ {e}
return T
```

   b. MAYBE-MST-B(G)

```
sort the edges into non-increasing order of edge weights w
T = E
for each edge e, in non-increasing order by weight
        if T – {e} is a connected graph
                T = T – {e}
return T
```

4. [Graphs – Interview question, 40 points]
Consider a 2D array of 1s and 0s of size NxN. The array is a "maze." You must start from the top left (index (0,0)) and find the length of the shortest path to the bottom right (index (n,n)). You may only travel across elements that have value "1". The top left and bottom right indices are guaranteed to be "1". You may only move horizontally or vertical one index with each step.

   For example, for the following input maze, a shortest path is along bolded numbers and has a length of 8:

```
11001
01011
01100
11110
11011
```

   Your program will accept an input file, called *maze.txt*, containing a maze. Your program should be executed with no inputs, read the provided maze from the file, determine N, the dimension of the maze, and output the length of the shortest path to the screen. If no path exists, your program should output '0'. You may use the provided *maze.txt* file to test your code.

   Submit your code, along with a modified *makefile* that can be used to compile and run your code on the lab computers.